CMPT 354 Mini-Project

Brandon Yang, Yecheng Wang

Step (2): Project Specifications

Library Item:

ItemID (Primary Key): Unique Identifier of the library item

Title: Name of the library item

Author: Name of the library item creator

Item_type: Book/ Magazine/ Journals/ CDs/ Records, etc.

Format: Print/ Online

Genre: Scientific/ Historical/ Romantic/ Military/ Political, etc.

Published Date: Date the library item is published, information for people

Availability: Available/ Borrowed/ Reserved/ Damaged

IsFutureItem: Boolean to determine if the item is reserved for future addition or not (Default: FALSE)

Restriction: Restricted item requires customers to meet the age limit to access (if applicable)

- **Checks:**
  - *Item_type* must be one of the defined types (Book, CD, DVD, Magazine, Journal, or Record)
  - *Format* must be one of the define formats (Print, Online, Audio, or Video)
  - *Availability* must be either: Available or Borrowed
  - *IsFutureItem* must be a boolean of 0 or 1
  - *Restrictions* must be >= 0 based on user age

Customers:

CustomerID (Primary Key): Unique Identifier of library customer

Date of Birth: Allow customer to access to restricted library items

Name: General customer information required

Address: General customer information required

Email: General customer information required

Phone Number: General customer information required

Preference: Type of library items they like, preferences help library events better find the targeted customers

Outstanding Fine Balance: Amount of overdue not returned books fines need to pay, refer to different Fine incidents.

- **Checks**:
  - *Outstanding_fine_balance* must be >= 0
  - *Email and phone* could be optional, *Email* is unique if provided

***Borrowing*:**

TransactionID (Primary Key): Unique Identifier for borrowing

ItemID (Foreign Key): The unique identifier of the library item borrowed

CustomerID (Foreign Key): The unique identifier of the customer borrowed the item

Borrowed Date: The date the item was borrowed

Due Date: The date the item needs to be returned

Returned Date: The date the item was actually returned

Amount of Fine: Fines for late return of items (if applicable)

- **Checks**:
  - *Amount_of_fine* must be >= 0

***Fines*:**

FineID (Primary Key): Unique Identifier for fine incidents

TransactionID (Foreign Key): The unique identifier of borrowing incident

CustomerID (Foreign Key): The unique identifier of customer related in the fine incident

Amount of Fine: Fines for late return of items

Fine Status: Paid / Unpaid

- **Checks**:
  - *Amount_of_fine* must be >= 0
  - *Fine_status* must be paid or unpaid

***Library Event*:**

EventID (Primary Key): Unique Identifier for library events

Event Name: Name of the event

Event Description: Brief description of the event

Event Type: book clubs, book related events, art shows, film screenings, etc.

Targeted Customers: Targeted customers for this event based on customer preferences

Restriction: Age limit for customers to meet in order to join the event

Location: Where the event is going to be held

Date & Time: When the event is going to be held

Capacity: Maximum number of people that can be accommodated in the activity

- **Checks**
  - *Capacity* and *restrictions* must be >= 0
  - *Event_type* must be of types (Workshop, Seminar, Film Screening, Book club, Art show)

Personnel

EmployeeID (Primary Key): Unique Identifier for personnels

Date of Birth: General personnel information required

Name: General personnel information required

Address: General personnel information required

Email: General personnel information required

Phone Number: General personnel information required

Salary: General personnel information required

Job Role: Role of the personnel

- **Checks:**
  - *Salary* must be >= 0
  - *Email* is unique if provided
  - *Job_role* is specified as "Volunteer" if user volunteers at the library

Register

RegisterID (Primary Key): Unique Identifier for event registration

EventID (Foreign Key): Identifier of the event

CustomerID (ForeignKey): Identifier of the registered customer

Manage

ManageID (Primary Key): Unique identifier for personnel management for events

EventID (Foreign Key): Identifier of the event

EmployeeID (ForeignKey): Identifier of the personnel in charge of the event

Assumptions

- Referential Integrity
  - All foreign keys in *Borrowing, fines, register,* and *manage* tables have ON DELETE CASCADE to ensure deletions would propagate and remove the corresponding tuples in the referenced relations.
- Fines are automatically calculated based on return date and borrow date, where fines begin accumulating 15 days after borrow date.

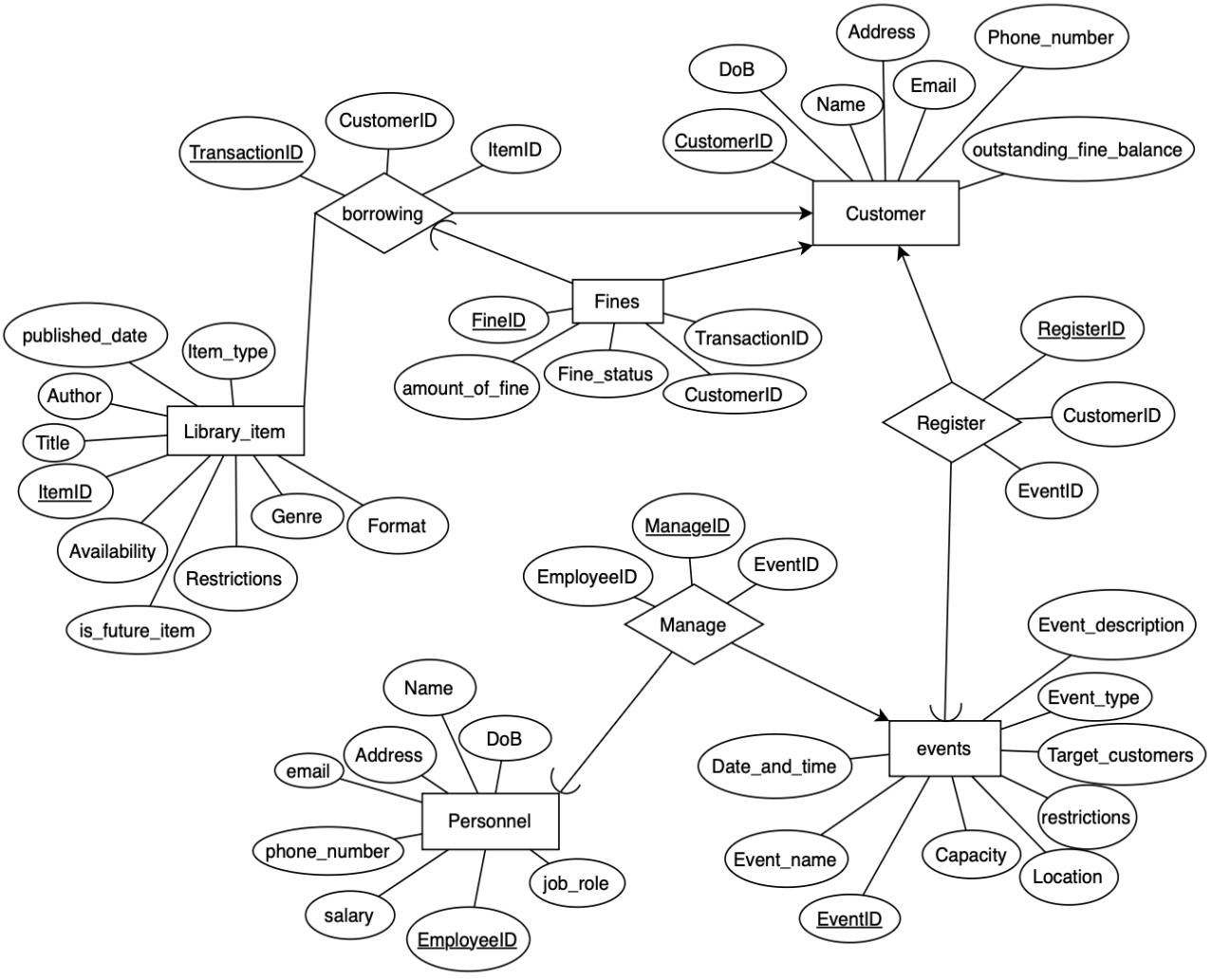
Triggers

- *Update_item_borrowed:* updates the item availability in *library_items* when an insert occurs on the *borrowing* table
- *Update_item_returned:* updates the item availability in *library_items* when item returned
- *Update_customer_fine_balance:* Trigger to update the *customers* table with the new fine balance when a fine is added into the *fines* table
- *Update_customer_fine_balance_paid:* update customer fine balance when fine is paid

Relationships

- Customers can borrow multiple items (One-to-Many between Customer and Borrowing)
- Each borrowing transaction involves one item (One-to-One between Borrowing and Library Item)
- Customers can register for multiple events (Many-to-Many between Customer and Library Event)
- Events are managed by one or more personnel (Many-to-Many between Personnel and Library Event)
- Fines are related to specific borrowing transactions (One-to-One between Fines and Borrowing)

Step (3): E/R Diagram



Step (4): Does your design allow anomalies?

Library Item FDs:

ItemID → {Title, Author, Type, Format, Genre, PublishedDate, Availability, IsFutureItem, Restriction}

ItemID uniquely determines all other attributes and is a super key, so Library Item table satisfy the BCNF.

If a book's author changes (e.g., correction or update), it is updated in one place instead of every borrowing record, this prevents Insert Anomalies

Customer FDs:

CustomerID → {Name, DateOfBirth, Address, Email, PhoneNumber, Preference, OutstandingFineBalance}

CustomerID uniquely determines all other attributes and is a super key, so Customer table satisfy the BCNF.

Borrowing FDs:

TransactionID → {ItemID, CustomerID, BorrowedDate, DueDate, ReturnedDate, AmountOfFine}

TransactionID uniquely determines all other attributes and is a super key, so Borrowing table satisfy the BCNF.

Fines FDs:

FineID → {TransactionID, CustomerID, AmountOfFine, FineStatus}

TransactionID → {CustomerID}

FineID uniquely determines all other attributes and is a super key, TransactionID does not but it is not a super key in Fine table, so Fines table satisfy the BCNF.

Library Event FDs:

EventID → {EventName, EventDescription, EventType, TargetedCustomers, Restriction, Location, DateTime, Capacity}

EventID uniquely determines all other attributes and is a super key, so Library Event table satisfy the BCNF.

Personnel FDs:

EmployeeID → {Name, DateOfBirth, Address, Email, PhoneNumber, Salary, Job Role}

EmployeeID uniquely determines all other attributes and is a super key, so Personnel table satisfy the BCNF.

Register FDs:

RegisterID→ {EventID, CustomerID}

RegisterID uniquely determines all other attributes and is a super key, so Registered Events table satisfy the BCNF.

Manage FDs:

ManageID → {EventID, EmployeeID}

ManageID uniquely determines all other attributes and is a candidate key, so Event Personnel table satisfy the BCNF.

Step (5): SQL Schema

```sql
CREATE TABLE IF NOT EXISTS customers (
    customer_id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    email TEXT UNIQUE,
    phone TEXT,
    dob TEXT NOT NULL,
    address TEXT,
    preferences TEXT,
    outstanding_fine_balance REAL DEFAULT 0.0 CHECK (outstanding_fine_balance >= 0)
)


CREATE TABLE IF NOT EXISTS library_items (
    item_id INTEGER PRIMARY KEY AUTOINCREMENT,
    title TEXT NOT NULL,
    author TEXT,
    item_type TEXT CHECK (item_type IN ('Book', 'CD', 'DVD', 'Magazine', 'Journal', 'Record')),
    format TEXT CHECK (format IN ('Print', 'Online', 'Audio', 'Video')),
    genre TEXT,
    published_date TEXT,
    availability TEXT DEFAULT 'Available' CHECK (availability IN ('Available', 'Borrowed')),
    is_future_item BOOLEAN DEFAULT 0 CHECK (is_future_item IN (0, 1)),
    restriction INTEGER DEFAULT 0 CHECK (restriction >= 0)
)


CREATE TABLE IF NOT EXISTS borrowing (
    transaction_id INTEGER PRIMARY KEY AUTOINCREMENT,
    item_id INTEGER NOT NULL,
    customer_id INTEGER NOT NULL,
    borrowed_date TEXT NOT NULL,
    due_date TEXT,
    returned_date TEXT,
    amount_of_fine REAL DEFAULT 0.0 CHECK (amount_of_fine >= 0),
    FOREIGN KEY (item_id) REFERENCES library_items(item_id) ON DELETE CASCADE,
    FOREIGN KEY (customer_id) REFERENCES customers(customer_id) ON DELETE CASCADE
)

CREATE TABLE IF NOT EXISTS fines (
    fine_id INTEGER PRIMARY KEY AUTOINCREMENT,
    transaction_id INTEGER NOT NULL,
    customer_id INTEGER NOT NULL,
    amount_of_fine REAL NOT NULL CHECK (amount_of_fine >= 0),
    fine_status TEXT DEFAULT 'Unpaid' CHECK (fine_status IN ('Paid', 'Unpaid')),
    FOREIGN KEY (transaction_id) REFERENCES borrowing(transaction_id) ON DELETE CASCADE,
    FOREIGN KEY (customer_id) REFERENCES customers(customer_id) ON DELETE CASCADE
)

CREATE TABLE IF NOT EXISTS events (
    event_id INTEGER PRIMARY KEY AUTOINCREMENT,
    event_name TEXT NOT NULL,
    event_description TEXT,
    event_type TEXT CHECK (event_type IN ('Workshop', 'Seminar', 'Film Screening', 'Book Club', 'Art Show')),
    targeted_customers TEXT,
    restriction INTEGER DEFAULT 0 CHECK (restriction >= 0),
    location TEXT,
    datetime TEXT NOT NULL,
    capacity INTEGER CHECK (capacity >= 0)
)
```

```sql
CREATE TABLE IF NOT EXISTS register (
    register_id INTEGER PRIMARY KEY AUTOINCREMENT,
    event_id INTEGER NOT NULL,
    customer_id INTEGER NOT NULL,
    FOREIGN KEY (event_id) REFERENCES events(event_id) ON DELETE CASCADE,
    FOREIGN KEY (customer_id) REFERENCES customers(customer_id) ON DELETE CASCADE
)

CREATE TABLE IF NOT EXISTS personnel (
    employee_id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    dob TEXT NOT NULL,
    address TEXT,
    email TEXT UNIQUE,
    phone TEXT,
    salary REAL DEFAULT 0.0 CHECK (salary >= 0),
    job_role TEXT NOT NULL
)

CREATE TABLE IF NOT EXISTS manage (
    manage_id INTEGER PRIMARY KEY AUTOINCREMENT,
    event_id INTEGER NOT NULL,
    employee_id INTEGER NOT NULL,
    FOREIGN KEY (event_id) REFERENCES events(event_id) ON DELETE CASCADE,
    FOREIGN KEY (employee_id) REFERENCES personnel(employee_id) ON DELETE CASCADE
)
```

Step (6): Populate Tables

```python
    # Populate customers table
    c.executemany('''
        INSERT INTO customers (name, email, phone, dob, address, preferences, outstanding_fine_balance)
        VALUES (?, ?, ?, ?, ?, ?, ?)
    ''', [
        ('Alice Johnson', 'alice@example.com', '123-456-7890', '1990-05-15', '123 Main St', 'Fiction', 0.0),
        ('Bob Smith', 'bob@example.com', '987-654-3210', '1985-08-22', '456 Elm St', 'Science', 5.0),
        ('Charlie Brown', 'charlie@example.com', '555-123-4567', '2000-01-10', '789 Oak St', 'History', 0.0),
        ('Diana Prince', 'diana@example.com', '444-555-6666', '1995-03-25', '321 Maple St', 'Fantasy', 2.5),
        ('Eve Adams', 'eve@example.com', '333-444-5555', '1992-07-18', '654 Pine St', 'Romance', 0.0),
        ('Frank Castle', 'frank@example.com', '222-333-4444', '1988-11-11', '987 Birch St', 'Military', 0.0),
        ('Grace Hopper', 'grace@example.com', '111-222-3333', '1991-09-09', '123 Cedar St', 'Education', 0.0),
        ('Hank Pym', 'hank@example.com', '999-888-7777', '1980-06-06', '456 Spruce St', 'Science', 0.0),
        ('Ivy League', 'ivy@example.com', '666-555-4444', '1993-03-03', '789 Willow St', 'Philosophy', 0.0),
        ('Jack Ryan', 'jack@example.com', '333-222-1111', '1987-12-12', '321 Aspen St', 'Thriller', 0.0)
    ])

    # Populate borrowing table
    c.executemany('''
        INSERT INTO borrowing (item_id, customer_id, borrowed_date, due_date, returned_date, amount_of_fine)
        VALUES (?, ?, ?, ?, ?, ?)
    ''', [
        (2, 1, '2025-03-01', '2025-03-15', None, 0.0),
        (3, 2, '2025-02-20', '2025-03-05', '2025-03-06', 1.0),
        (1, 3, '2025-03-10', '2025-03-24', None, 0.0),
        (4, 4, '2025-03-12', '2025-03-26', None, 0.0),
        (5, 5, '2025-03-15', '2025-03-29', None, 0.0),
        (6, 6, '2025-03-20', '2025-04-03', None, 0.0),
        (7, 7, '2025-03-22', '2025-04-05', None, 0.0),
        (8, 8, '2025-03-25', '2025-04-08', None, 0.0),
        (9, 9, '2025-03-27', '2025-04-10', None, 0.0),
        (10, 10, '2025-03-30', '2025-04-13', None, 0.0)
    ])
```

```
        # Populate library_items table
    c.executemany('''
        INSERT INTO library_items (title, author, item_type, format, genre, published_date, availability, is_future_item, restriction)
        VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)
    ''', [
        ('The Great Gatsby', 'F. Scott Fitzgerald', 'Book', 'Print', 'Fiction', '1925-04-10', 'Available', 0, 0),
        ('1984', 'George Orwell', 'Book', 'Print', 'Dystopian', '1949-06-08', 'Borrowed', 0, 0),
        ('To Kill a Mockingbird', 'Harper Lee', 'Book', 'Print', 'Fiction', '1960-07-11', 'Available', 0, 0),
        ('The Catcher in the Rye', 'J.D. Salinger', 'Book', 'Print', 'Fiction', '1951-07-16', 'Available', 0, 0),
        ('The Beatles: Abbey Road', 'The Beatles', 'CD', 'Audio', 'Music', '1969-09-26', 'Available', 0, 0),
        ('The Matrix', 'Wachowski Brothers', 'DVD', 'Video', 'Sci-Fi', '1999-03-31', 'Available', 0, 13),
        ('National Geographic', 'Various', 'Magazine', 'Print', 'Science', '2025-01-01', 'Available', 1, 0),
        ('Python Programming', 'Guido van Rossum', 'Book', 'Print', 'Education', '2010-05-01', 'Available', 0, 0),
        ('The Art of War', 'Sun Tzu', 'Book', 'Print', 'Philosophy', '500 BC', 'Available', 0, 0),
        ('The Hobbit', 'J.R.R. Tolkien', 'Book', 'Print', 'Fantasy', '1937-09-21', 'Available', 1, 0)
    ])


    # Populate events table
    c.executemany('''
        INSERT INTO events (event_name, event_description, event_type, targeted_customers, restriction, location, datetime, capacity)
        VALUES (?, ?, ?, ?, ?, ?, ?, ?)
    ''', [
        ('Book Club', 'Discussing the latest bestsellers.', 'Book Club', 'Adults', 0, 'Room A', '2025-03-25 18:00', 20),
        ('Art Show', 'Exhibition of local artists.', 'Art Show', 'All Ages', 0, 'Room B', '2025-03-30 14:00', 50),
        ('Film Screening', 'Classic movie night.', 'Film Screening', 'Teens', 13, 'Room C', '2025-04-05 19:00', 30),
        ('Coding Workshop', 'Learn Python programming.', 'Workshop', 'Adults', 0, 'Room D', '2025-04-10 10:00', 15),
        ('Story Time', 'Reading stories for kids.', 'Book Club', 'Kids', 0, 'Room E', '2025-04-15 11:00', 25),
        ('Poetry Night', 'Share and listen to poetry.', 'Seminar', 'Adults', 0, 'Room F', '2025-04-20 19:00', 30),
        ('Chess Tournament', 'Compete in a chess tournament.', 'Workshop', 'All Ages', 0, 'Room G', '2025-04-25 14:00', 20),
        ('Photography Workshop', 'Learn photography basics.', 'Workshop', 'Teens', 13, 'Room H', '2025-04-30 10:00', 15),
        ('Music Night', 'Live music performances.', 'Art Show', 'All Ages', 0, 'Room I', '2025-05-05 18:00', 40),
        ('Science Fair', 'Explore science projects.', 'Seminar', 'Kids', 0, 'Room J', '2025-05-10 12:00', 50)
    ])
```

```
    # Populate register table
    c.executemany('''
        INSERT INTO register (event_id, customer_id)
        VALUES (?, ?)
    ''', [
        (1, 1), (2, 2), (3, 3), (4, 4), (5, 5),
        (6, 6), (7, 7), (8, 8), (9, 9), (10, 10)
    ])

    # Populate personnel table
    c.executemany('''
        INSERT INTO personnel (name, dob, address, email, phone, salary, job_role)
        VALUES (?, ?, ?, ?, ?, ?, ?)
    ''', [
        ('John Doe', '1980-01-01', '123 Library St', 'john.doe@example.com', '123-456-7890', 50000, 'Librarian'),
        ('Jane Smith', '1990-02-15', '456 Library St', 'jane.smith@example.com', '987-654-3210', 45000, 'Assistant Librarian'),
        ('Mark Johnson', '1985-03-10', '789 Library St', 'mark.johnson@example.com', '555-123-4567', 0, 'Volunteer'),
        ('Emily Davis', '1995-07-20', '321 Library St', 'emily.davis@example.com', '444-555-6666', 0, 'Volunteer'),
        ('Michael Brown', '1988-11-11', '654 Library St', 'michael.brown@example.com', '333-444-5555', 60000, 'Manager'),
        ('Sarah Connor', '1982-05-12', '987 Library St', 'sarah.connor@example.com', '222-333-4444', 55000, 'Event Coordinator'),
        ('Tom Hardy', '1993-08-25', '654 Library St', 'tom.hardy@example.com', '111-222-3333', 0, 'Volunteer'),
        ('Anna Bell', '1996-09-15', '321 Library St', 'anna.bell@example.com', '999-888-7777', 0, 'Volunteer'),
        ('Chris Evans', '1987-07-04', '123 Library St', 'chris.evans@example.com', '666-555-4444', 70000, 'Director'),
        ('Natalie Portman', '1992-03-22', '456 Library St', 'natalie.portman@example.com', '333-222-1111', 0, 'Volunteer')
    ])

    # Populate manage table
    c.executemany('''
        INSERT INTO manage (event_id, employee_id)
        VALUES (?, ?)
    ''', [
        (1, 1), (2, 2), (3, 3), (4, 4), (5, 5),
        (6, 6), (7, 7), (8, 8), (9, 9), (10, 10)
    ])
```

```
    # Populate fines table
    c.executemany('''
        INSERT INTO fines (transaction_id, customer_id, amount_of_fine, fine_status)
        VALUES (?, ?, ?, ?)
    ''', [
        (1, 1, 0.0, 'Paid'), (2, 2, 1.0, 'Paid'), (3, 3, 2.5, 'Unpaid'),
        (4, 4, 0.0, 'Paid'), (5, 5, 0.0, 'Paid'), (6, 6, 0.0, 'Paid'),
        (7, 7, 0.0, 'Paid'), (8, 8, 0.0, 'Paid'), (9, 9, 0.0, 'Paid'),
        (10, 10, 0.0, 'Paid')
    ])
```